SOFA: A modular yet efficient simulation framework

François Faure, Jérémie Allard, Stéphane Cotin, Paul Neumann, Pierre-Jean Bensoussan, Christian Duriez, Hervé Delingette, Laurent Grisoni

SOFA: A MODULAR YET EFFICIENT SIMULATION FRAMEWORK

F. FAURE, J. ALLARD[+], S. COTIN[*], P. NEUMANN[+], P-J. BENSOUSSAN*, C. DURIEZ*, H. DELINGETTE*, L. GRISONI**

* INRIA – Evasion, Alcove, and Asclepios teams – FRANCE
[+] CIMIT Sim Group – 65 Lansdowne Street – Cambridge, MA 02139 – USA

Abstract. SOFA is a new open source framework primarily targeted at medical simulation research and industry. It is based on a scene graph data structure extended to physical models and abstract algorithms. Additionally, multiple models of the same objects can easily be used to optimize different tasks such as force computation, collision handling, and rendering. This results in a highly flexible architecture able to model and animate a wide range of simulated objects. We explain the main concepts of SOFA and detail an example of application to a surgery procedure.

1 Introduction

Computer-based training systems offer an elegant solution to the current need for better training in Medicine. It is widely admitted that surgical education and planning can highly benefit from computer simulations. However in spite of the impressive developments in the field of medical simulation, some fundamental problems still hinder the acceptance of this valuable technology in daily clinical practice. In particular, the multi-disciplinary aspect of medical simulation requires the integration within a single environment of leading-edge solutions in areas as diverse as visualization, biomechanical modeling, haptics or contact modeling. This diversity of problems makes it challenging for researchers to make progress in specific areas, and leads rather often to duplication of efforts.

For the past few years, there have been attempts at designing software toolkits for medical simulation such as SPRING [8], GiPSi [4], VRASS [3], or SSTML [2]. Although these different solutions had the same aim to provide an open source answer to the various challenges of medical simulation research and development, they were generally limited by their organization or had restrictions on the range of physical models, such as mass-spring systems, with a limited choice of algorithms for time integration and collision detection. We propose a different approach through a very modular and flexible

software framework called SOFA [1]. This open source framework allows independently developed algorithms to interact together within a common simulation while minimizing the development time required for integration.

The main objectives of the SOFA framework are:

- Provide a common software framework for the medical simulation community
- Enable component sharing / exchange and reduce development time
- Promote collaboration among research groups
- Enable validation and comparison of new algorithms
- Help standardize the description of anatomical and biomechanical datasets

Our main overall goal is to develop a flexible framework while minimizing the impact of this flexibility on the computation overhead. To achieve these objectives, we have developed a new architecture that implements a series of concepts described below.

2 The SOFA architecture

2.1 Mechanical scene graph

Implementing the physical simulation of a given model is generally a hard task. You first describe the model using physical entities such as points, masses, forces. You then implement physical equations and algorithms to animate the model. And then, you finally display images using computer graphics tools.

Scene graphs are popular graphics tools because they allow you to represent the model by instantiating simple software objects using your data, and then a generic rendering engine does the job of displaying images. The objects are organized in tree-like structures. Some of them store coordinates, while others represent shapes such as polygon meshes, light sources, materials, or textures. The objects have various attributes such as position, color or file name. Once the objects and attributes are set, the standard graphics operations such as polygon rasterization, pixel shading, transparency accumulation or hidden parts removal, are automatically implemented by the rendering engine. Using scene graphs, a limited know-ledge of computer graphics is sufficient to generate beautiful images of complex models. They are widely used in simulators.

The aim of SOFA is to bring the power of scene graphs to mechanical simulations. It extends traditional scene graphs with physical components such as force fields, masses, and constraints. Some other components represent physics algorithms. Once a model, along with its associated algorithms, is expressed as a SOFA *mechanical scene graph*, standard physical algorithms are automatically implemented and available. This includes various explicit and implicit time integration methods as well as collision detection and reaction between various shapes. SOFA is of course extendable using new components.

In this paper, we focus on the simulation of viscoelastic bodies. Most explicit and implicit time integration methods can be decomposed into a few physical procedures: given positions and velocities, accumulate forces; given forces, compute accelerations; filter out forbidden displacements; compute the product of the

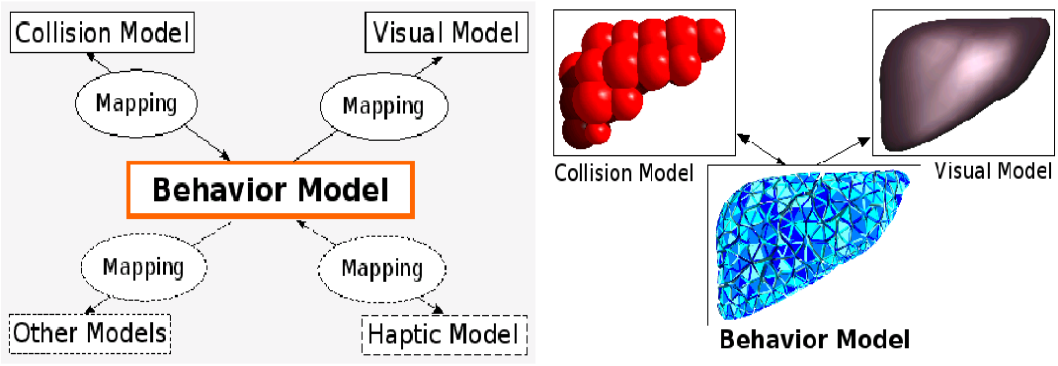

Fig. 1:
Multimodels in SOFA. Left: possible representations for a simulated object, with the Behavior Model controlling the update of the other representations through a series of mappings. Right: examples of these representations for a liver model. Notice how the Visual Model is more detailed than the Behavior Model and how the Collision Model relies on a very different representation.

mass and stiffness matrices with a vector. The SOFA engine implements these procedures by traversing the scene graph using visitors which trigger the appropriate methods of the components. This guarantees that the procedures are automatically implemented, provided that the scene graph is syntactically correct.

SOFA can be used in different ways. Its default application allows you to model scenes composed of various rigid, viscoelastic or fluid bodies in contact. You interact directly with the scene by picking and pulling the objects. For a given model, you can compare time integration or collision detection methods by simply replacing components. For a given shape, you can try various mechanical models including mass-springs and FEM. To implement a sophisticated simulation, you can use the available standard components and develop only the components related to your area of expertise. Section 3 presents such a case.

2.2 Multi-model representation

Any simulation involves, to some extent, the computation of visual feedback, haptic feedback, and interactions between medical devices and anatomical structures. This typically translates into a simulation loop where, at each time step, collisions between objects are detected, deformation and collision response are computed, and the resulting state can be visually and haptically rendered. To perform each of these actions, the various algorithms involved in the simulation rely implicitly on different data structures for the simulated objects. In SOFA, we explicitly decompose an object into various representations, in such a way that each representation is more suited toward a particular task – rendering, deformation, or collision detection. Then, these representations are linked together so they can be coherently updated. We call the link between these representations a *mapping*. Various mapping functions can be defined, and each mapping will associate a set of primitives of a representation to a set of

primitives in the other representation (see Figure 1). For instance, a mapping can connect degrees of freedom in a Behavior Model to vertices in a Visual Model.

3 Application to eye surgery

In this section, we show and discuss a SOFA scene used to simulate an ophthalmology procedure called a vitrectomy which reattaches the retina. In some diabetic patients, proliferative fibrovascular tissue growth can lead to traction which can disrupt the retina's nature cohesion. To model a diabetic circumferential traction case, the circular fibovascular membrane or *scar tissue*, compresses the retina radially. Once released, the retina relaxes to its natural shape.

Figure 2 illustrates the physical model. The scar tissue, in blue, and the retina, in green, are modeled using a mass-spring system. Fixed particles, circled in black, mimic the retina's natural attachments. The short springs, in red, are removed when the blade touches one of their endpoints.

The scene is decomposed in three main parts, as illustrated in figure 3: the physical model of the eye, the tool manipulated by the surgeon, and the fixed objects. The degrees of freedom (DOF) of the tool are stored in a MechanicalObject component. They are controlled by the haptics device and optionally recording or replaying the trajectory from a file using the WriteState and ReadState components. Two visible objects are attached to these DOF using mappings. One represents the handle and the other represents the tip of the tool. To allow collision detection and modeling of the tool tip with other scene elements, we use a LineModel, which defines a set of collidable lines based on an associated mesh topology. The associated DOF (the edge endpoints) are mapped from the rigid body DOF.

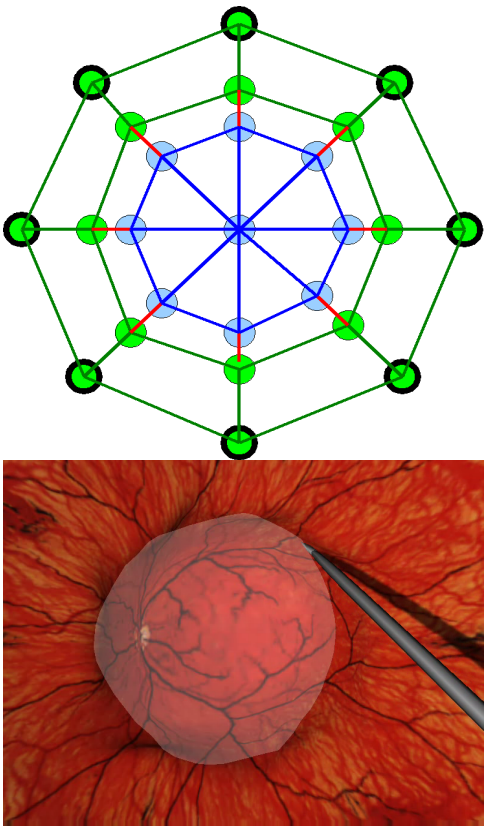

Fig. 2:
Top: the physical model of the retina and scar tissue. Bottom: a simulation snapshot.

The deformable object is a physical model controlled by an implicit integration solver implemented in the CgImplicitSolver component. Its masses and fixed-point constraints are given in the DiagonalMass and FixedConstraint components, respectively. Two sets of springs are used. The Retina springs model the behavior of the retina and scar

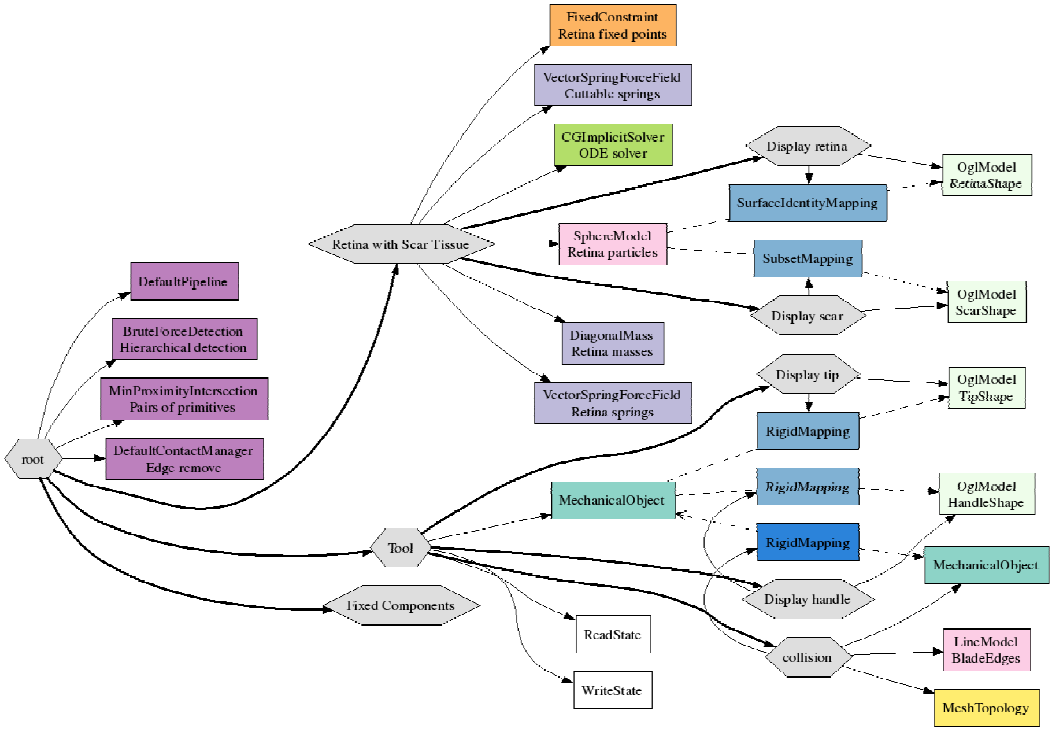

Fig. 3:
The scene graph of the vitrectomy scene (fixed objects omitted).

tissue, while the Cuttable springs model the springs which can be removed by cutting. Component SphereModel models the DOF as particles with associated spheres. Two drawable shapes are associated with the retina, one for the standard retina and one for the scar tissue. Mappings are used to update the positions of the shape vertices based on retina DOF.

Collision detection and handling is organized as a pipeline including a hierarchical detection which implements broad-phase and narrow-phase detection which prunes the possible pair of intersecting primitives, followed by exact primitive intersection phase, as illustrated in figure 4. Detected collisions are then processed by contact creation and response phase.

Each part of the pipeline is implemented in a component and can be customized for special purposes. The overall pipeline is managed by the Default-Pipeline component, which defers to the Hierarchical detection component to implement hierarchical bounding volume strategy and the MinProximity-Intersection component for precise detection between pairs of primitives. When a collision is detected, it is handled by the contact manager.

This simulation was originally implemented as a stand-alone custom application. Its port to SOFA mainly uses standard components and runs at

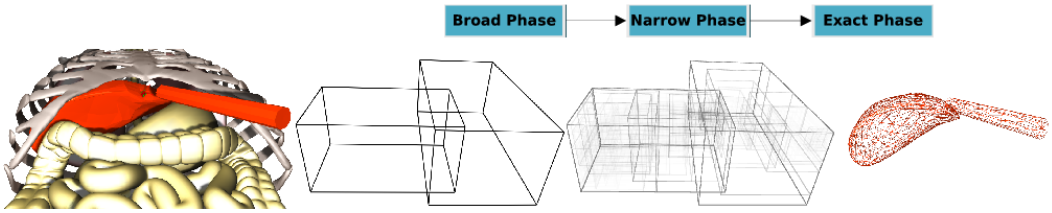

Fig. 4:
The SOFA collision pipeline.

the same speed. Two new components corresponding to the specific technical contributions of this work where derived. The first implements a custom spring model, while the other reacts to contact by removing springs instead of setting up reaction forces as done by the default response strategy.

3.1 Efficiency

Most of the computation time in physical simulation is spent in matrix (mass and stiffness) computations and collision detection. SOFA matrices are inherently sparse due to the tree data structure, and matrix-vector products are performed in linear time. This allows us to efficiently implement explicit and implicit time integrators [3]. Collision detection uses standard hierarchical bounding volumes to cull out unnecessary tests on geometric primitives. This makes the efficiency of SOFA comparable with state-of-the-art general purpose implementations.

Moreover, we are implementing strategies to automatically perform parallel computations on the GPU. We have developed an early prototype of a laparoscopic simulation system in which the liver and intestines are modelled as deformable models which can be manipulated using a laparoscopic instrument and can collide with the ribs, as illustrated in Figure 4. We use this scene as a benchmark for various models and algorithms. The modularity of the SOFA architecture allows us to easily experiment different constitutive models for the organs such as a co-rotational FEM and spring-based FFD grids. To find the right balance between performances and accuracy, we tested both FEM and spring-based models of varying resolution on the liver (see Fig. 5). While FEM-based models are generally more accurate, spring-based FFD grids are easier to set up and to change the resolution, as only the surface mesh is required.

CPU-based models Both FEM and springs were able to simulate the liver at interactive rates when executed on the CPU. Using FEM models consisting of 1800 and 30000 tetrahedra, the simulation achieved 65 and 4 iterations per second, respectively. Using springs grids of sizes 17x8x10, 20x10x12 and 24x13x15 the simulation achieved respectively 110, 67, and 37 iterations per second.

GPU-based models SOFA currently supports transfering part of the simulations to the GPU thanks to the NVIDIA CUDA library. It is still in its early stage, hence only springs-based models are currently supported. However, it already achieves good

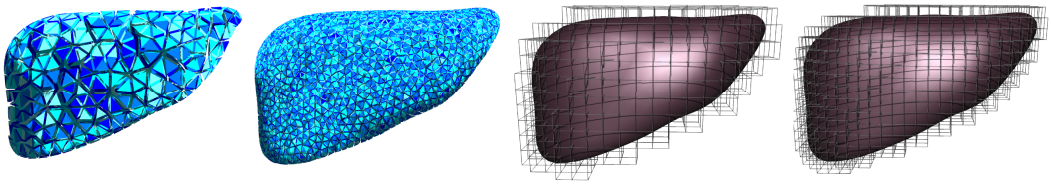

Fig. 5:
FEM-based and spring grid-based behavior models at multiple resolutions

speedups compared to the CPU implementation. In the case of the liver, we where able to simulate the same springs grids models of sizes 17x8x10, 20x10x12 and 24x13x15 at respectively 261, 227, and 178 iterations per second. Simulating the liver at this speed is useful as it frees the CPU for other parts of the simulation, and opens new possibility to interactively support higher precisions models.

Conclusion

We have discussed the main concepts of SOFA and presented its application to a real surgical simulation. Compared with the implementation of a stand-alone application, using SOFA allows one to re-use and compare a wide variety of available models and algorithms, while focusing on one's specific area of expertise. The efficiency is comparable with the best implementations, and parallel processing using the GPU will be increasingly available.

SOFA is freely available and its user community is growing.

In future work, we plan to add support for high-frequency haptics feedback, more collision detection and modeling algorithms, dense matrix computations, and more complex cuts.

BIBLIOGRAPHY

[1] ALLARD J. *ET. AL.*
SOFA - an Open Source Framework for Medical Simulation.
www.sofa-framework.org
PROCEEDINGS OF MMVR, 2007.

[2] BACON J., TARDELLA N., PRATT J., ENGLISH J.
The Surgical Simulation and Training Markup Language: An XML-Based Language for Medical Simulation
Proceedings of MMVR, 2006, 37–42

[3] BARAFF D., WITKIN A.
Large steps in cloth simulation
Proceedings of SIGGRAPH, 1998

[4] GOKTEKIN T., CAVUSOGLU M.C., TENDICK F.
Gipsi: An open source software development framework for surgical simulation
International Symposium on Medical Simulation, 2004, 240–248

[5] KAWASAKI M. *ET AL.*
VRASS (Virtual Reality Aided Simulation)
www.kuhp.kyoto-u.ac.jp/ mi/ research/vrass/index_en.shtml

[6] MULLER M., GROSS M.
Interactive virtual materials
Graphics Interface, 2004, 239–246

[7] NESME M., PAYAN Y., FAURE F.
Efficient, physically plausible finite elements
Eurographics, 2005

[8] K. MONTGOMERY *ET. AL.*
Spring: A general framework for collaborative, real-time surgical simulation.
PROCEEDINGS OF MMVR, 2002, 23–26.